


# RMARKDOWN



Prise en main et un peu plus

New Project

[Back](#) **Create New Project**

The R logo, a blue cube with a white 'R' on it.

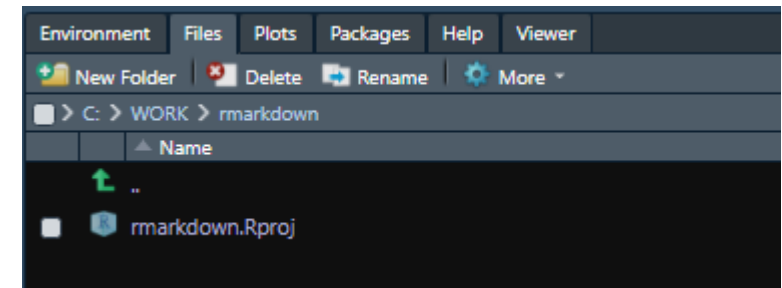
Directory name:

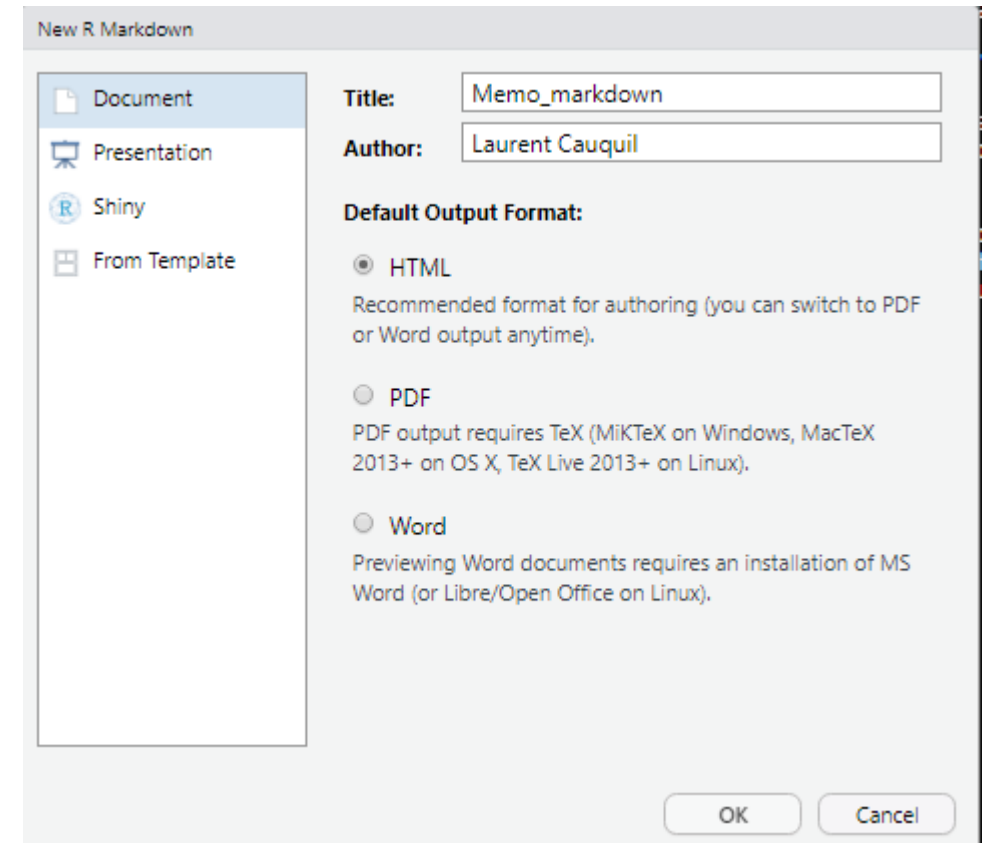
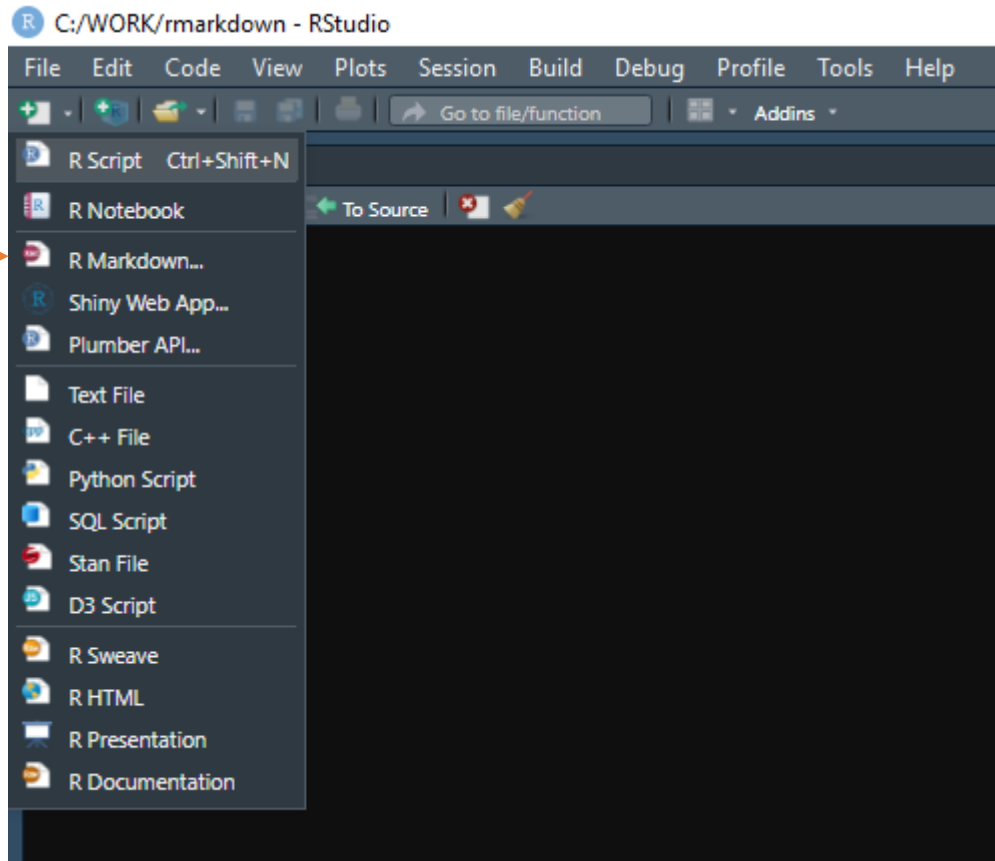
Create project as subdirectory of:  
 [Browse...](#)

☐ Create a git repository

☐ Open in new session

[Create Project](#) [Cancel](#)



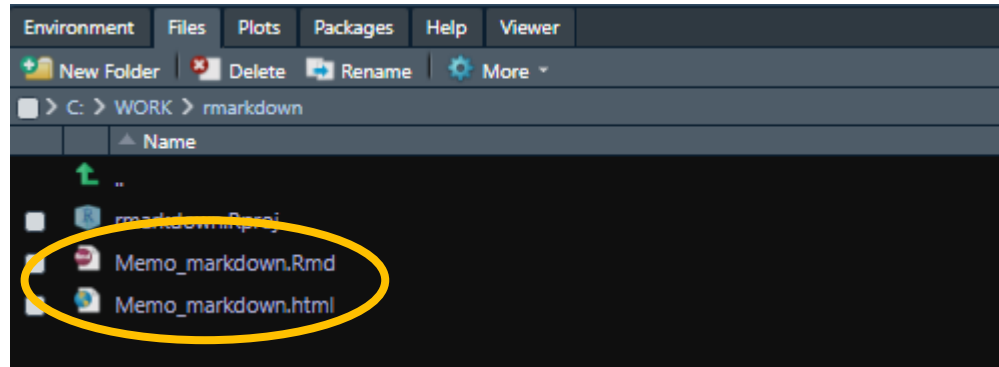


```
Untitled1
1 ---
2 title: "Memo_markdown"
3 author: "Laurent Cauquil"
4 date: "07/10/2019"
5 output: html_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more
15 details on using R Markdown see <http://rmarkdown.rstudio.com>.
16
17 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R
18 code chunks within the document. You can embed an R code chunk like this:
19
20 ```{r cars}
21 summary(cars)
22 ```
23
24 ## Including Plots
25
26 You can also embed plots, for example:
27
28 ```{r pressure, echo=FALSE}
29 plot(pressure)
30 ```
31
32 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

Ctrl + shift + k

A screenshot of the RStudio interface. The top toolbar shows the 'Knit' button, which is highlighted by an orange arrow. A dropdown menu is open, showing options: 'Knit to HTML', 'Knit to PDF', 'Knit to Word', 'Knit with Parameters...', 'Knit Directory', and 'Clear Knitr Cache...'. The main editor window displays an R Markdown document. The document content includes a title '## R Markdown', a paragraph about R Markdown, an R code chunk for 'summary(cars)', another paragraph '## Including Plots', a paragraph about embedding plots, a second R code chunk for 'plot(pressure)' with 'echo=FALSE', and a final paragraph explaining the 'echo = FALSE' parameter. The code chunks are enclosed in triple backticks with the language 'r' specified.

```
1  ---  
2  title: "R Markdown"  
3  author: "John Fox"  
4  date: "2020-01-01"  
5  output: "html_document"  
6  ---  
7  
8  knitr: {  
9    echo = TRUE  
10  }  
11  
12  ## R Markdown  
13  
14  This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.  
15  
16  When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:  
17  
18  ```{r cars}  
19  summary(cars)  
20  ```  
21  
22  ## Including Plots  
23  
24  You can also embed plots, for example:  
25  
26  ```{r pressure, echo=FALSE}  
27  plot(pressure)  
28  ```  
29  
30  Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.  
31
```



# Memo\_markdown

Laurent Cauquil

30/09/2019

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

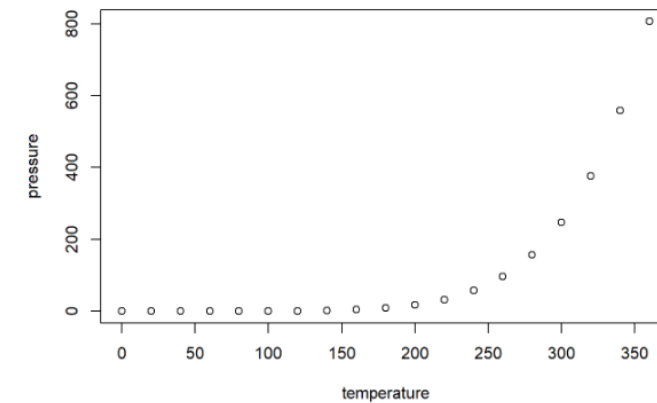
When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed      dist
##  Min.   : 4.0   Min.   : 2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

## Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

# Pourquoi utiliser R Markdown ?

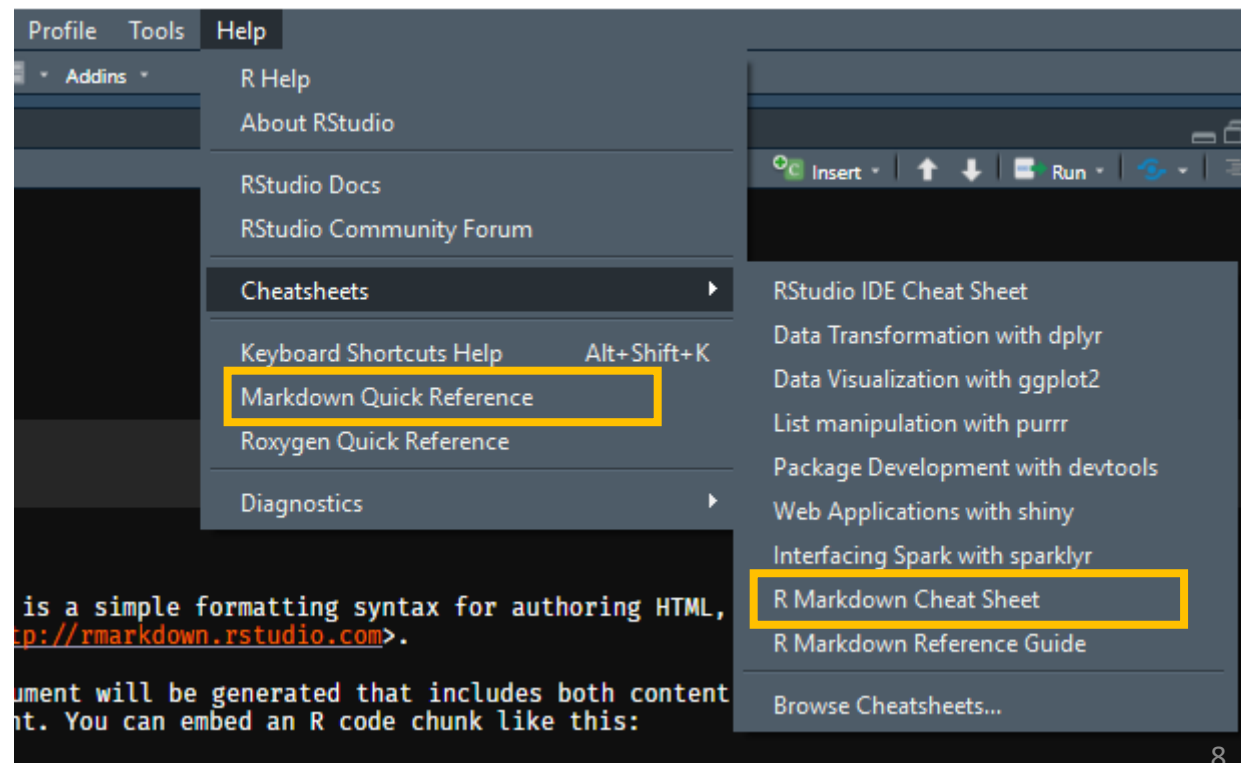
- Permet de produire des documents scientifiques mélangeant du texte, du code R et le résultat de l'exécution de ce code de manière automatique
- Permet de regrouper tous les résultats d'analyses y compris les graphes en un seul document
- Génère des rapports dans différents formats (HTML, PDF, Office WORD,...)

Facile à prendre en main au début, les utilisateurs avancés peuvent par la suite customiser leur document à volonté via du code HTML et CSS

# Documentation



- La documentation officielle se trouve à cette page [R Markdown: The Definitive Guide](#) (exemple de document R Markdown !)
- Ne pas oublier l'incontournable [cheatsheet](#) de RStudio
- De plus, il existe une multitude de tutoriels disponibles en ligne y compris en français (Cf. annexes)







Pour les documents PDF, il faut veiller à installer une installation fonctionnelle de LaTeX

Il est recommandé d'utiliser tinyTeX (<https://yihui.name/tinytex/>)

```
install.packages("tinytex")  
tinytex::install_tinytex() # install tinyTeX
```



Certains affichages ne sont pas compatibles avec un document PDF, notamment les tables et les graphes interactifs

# Anatomie d'un R Markdown

Un document R Markdown est constitué de 3 parties:

- des méta-données (obligatoires)
- du texte
- du code à exécuter (chunk)

```
# insertion d'un chunk dans le script

```{r}
x = 2 + 2
x
```
```

# Méta-données: YAML (Yet Another Markup Language)

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output:  
  html_document:  
    toc: TRUE  
    fig_width: 9  
---
```

## Méta-données: YAML

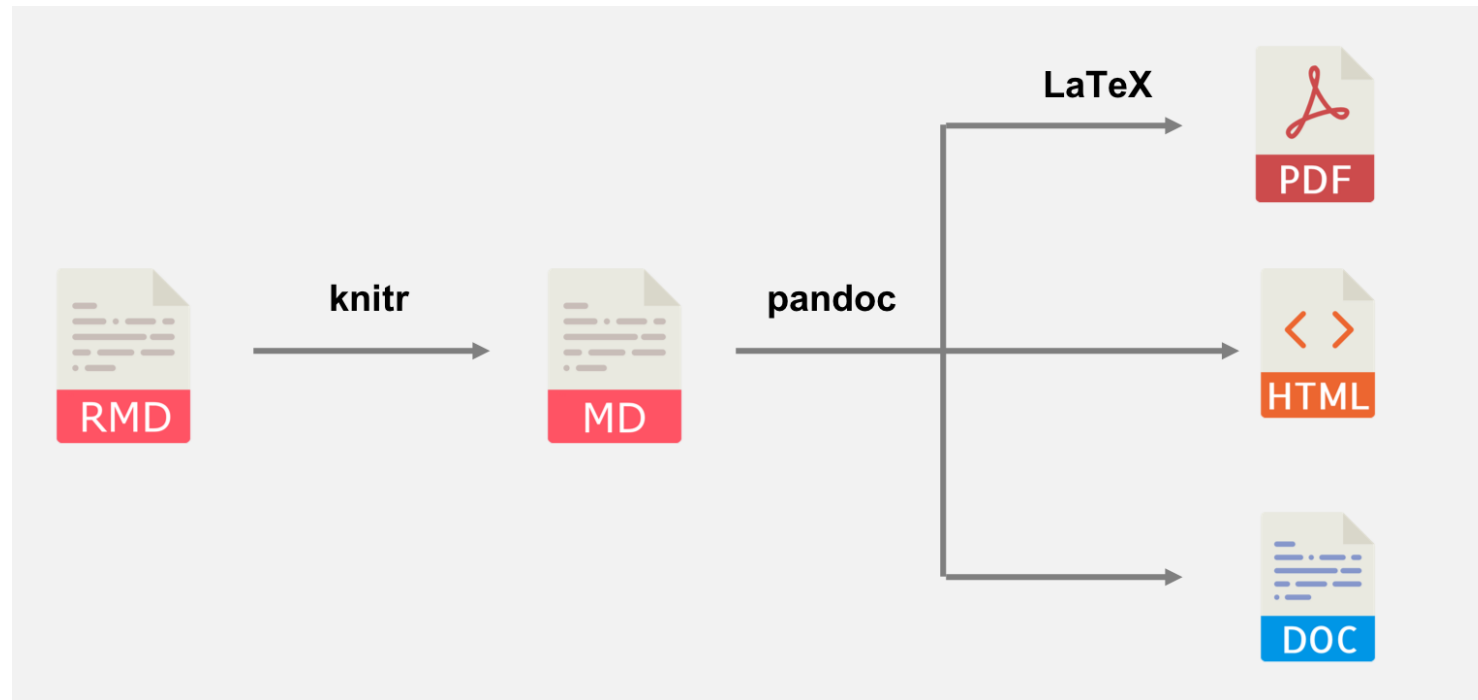
Partie la plus obscure du script quand on démarre

- Les méta-données sont saisies en début de script entre une paire de 3 tirets ---
- Elles renseignent certains paramètres qui permettent de contrôler des caractéristiques du document en fonction du format de sortie
- La syntaxe utilisée suit le format YAML (<https://fr.wikipedia.org/wiki/YAML> )

Contrôle du rendu général du document

# Méta-données: YAML

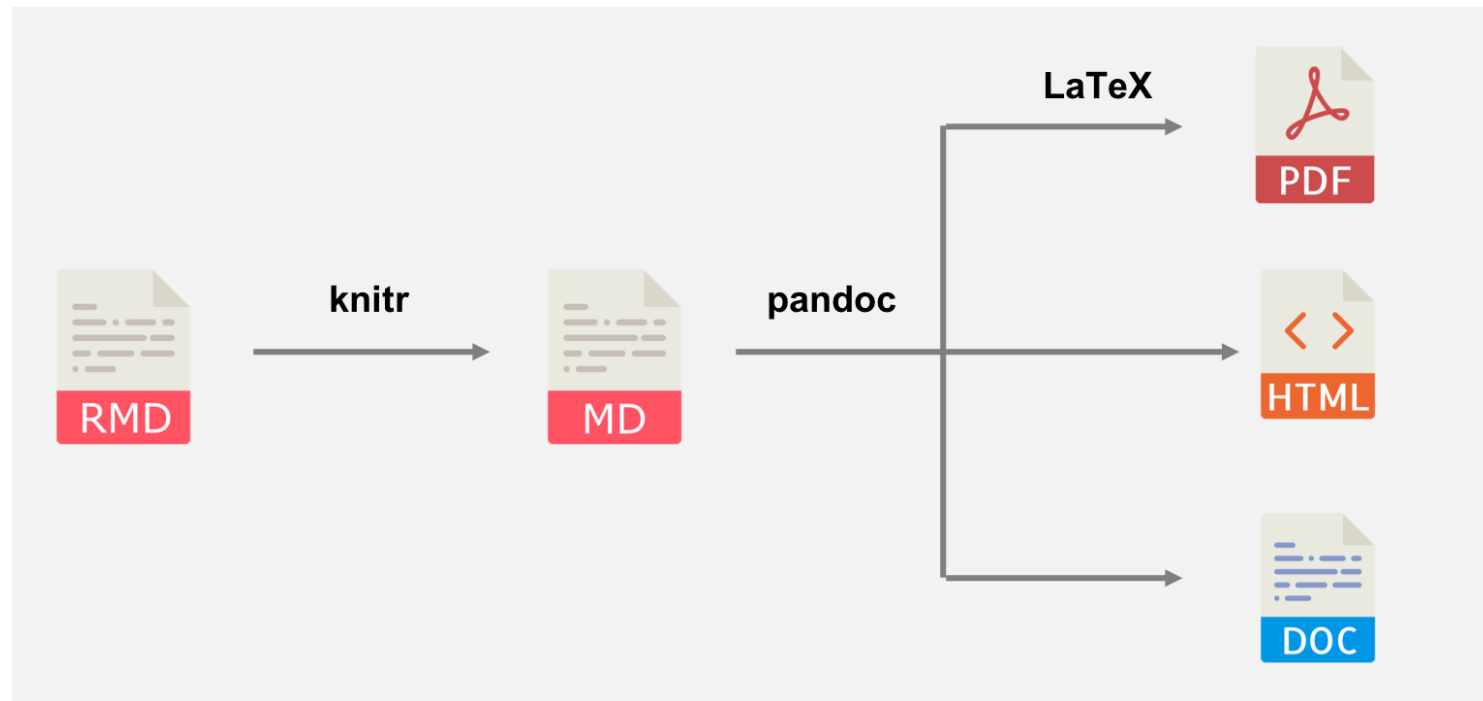
## Processus de création du document



# Méta-données: YAML

Pour “contrôler” le rendu de notre document on intervient:

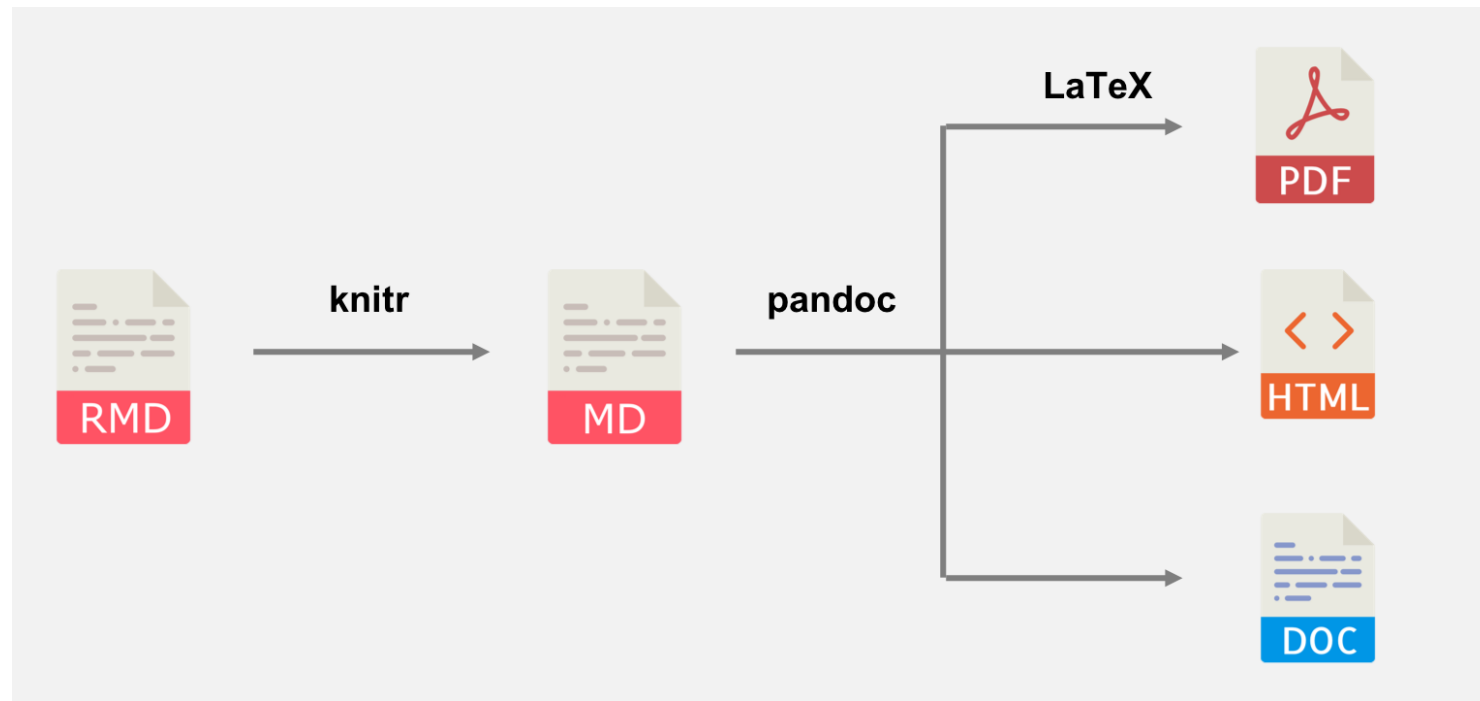
- Au niveau de la traduction du fichier `.rmd` en `.md` avec des options de **knitr**
- Au niveau de la conversion dans le format de sortie en passant des informations à **pandoc**



## Méta-données: YAML

Les options de knitr sont précisées dans les chunks (code R), elles s'appliquent:

- Soit uniquement au chunk dans lequel elles se trouvent
- Soit à tout le document



## Méta-données: YAML

Les méta-données du YAML permettent de préciser le rendu final du document

➤ titre, la présence d'un sommaire, d'une bibliographie...

Les méta-données dépendent donc de **pandoc** et des options disponibles en fonction du format de sortie

Pour maîtriser cette partie il est "conseillé de lire" la doc de Pandoc (ce qui n'est pas vraiment la partie la plus fun)

Ou bien on s'inspire de ce qui existe déjà et de l'aide !



## Méta-données: YAML

A l'ouverture d'un nouveau markdown .Rmd un YAML de base est généré par défaut

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output: html_document  
---
```

Dans le cas suivant, on peut renseigner le titre, l'auteur, la date et le format de sortie

## Méta-données: YAML

On “knite”,  
le document produit est une page HTML placée dans le répertoire où se trouve le script

Elle a le même nom que le script mais avec l’extension HTML

Un fichier **.md** est également généré lors de la conversion mais par défaut il n'est pas gardé

# Comment personnaliser ?

>?rmarkdown::html\_document

>?rmarkdown::pdf\_document

```
html_document {rmarkdown}          R Documentation
Convert to an HTML document
Description
Format for converting from R Markdown to an HTML document.
```

## Usage

```
html_document(toc = FALSE, toc_depth = 3, toc_float = FALSE,
  number_sections = FALSE, section_divs = TRUE, fig_width = 7,
  fig_height = 5, fig_retina = 2, fig_caption = TRUE, dev = "png",
  df_print = "default", code_folding = c("none", "show", "hide"),
  code_download = FALSE, smart = TRUE, self_contained = TRUE,
  theme = "default", highlight = "default", mathjax = "default",
  template = "default", extra_dependencies = NULL, css = NULL,
  includes = NULL, keep_md = FALSE, lib_dir = NULL,
  md_extensions = NULL, pandoc_args = NULL, ...)
```

# Tableau dans cheatsheets



| sub-option            | description   | html | pdf | word | odt | rtf | md | github | ioslides | slidy | beamer |
|-----------------------|---|------|-----|------|-----|-----|----|--------|----------|-------|--------|
| citation_package      | The LaTeX package to process citations, natbib, biblatex or none                |      | X   |      |     |     | X  |        |          |       | X      |
| code_folding          | Let readers to toggle the display of R code, "none", "hide", or "show"          | X    |     |      |     |     |    |        |          |       |        |
| colortheme            | Beamer color theme to use   |      |     |      |     |     |    |        |          |       | X      |
| css                   | CSS file to use to style document   | X    |     |      |     |     |    |        | X        | X     |        |
| dev                   | Graphics device to use for figure output (e.g. "png")                           | X    | X   |      |     |     | X  | X      | X        | X     | X      |
| duration              | Add a countdown timer (in minutes) to footer of slides                          |      |     |      |     |     |    |        |          | X     |        |
| fig_caption           | Should figures be rendered with captions?                                       | X    | X   | X    | X   |     |    |        | X        | X     | X      |
| fig_height, fig_width | Default figure height and width (in inches) for document                        | X    | X   | X    | X   | X   | X  | X      | X        | X     | X      |
| highlight             | Syntax highlighting: "tango", "pygments", "kate", "zenburn", "textmate"         | X    | X   | X    |     |     |    |        |          | X     | X      |
| includes              | File of content to place in document (in_header, before_body, after_body)       | X    | X   |      | X   |     | X  | X      | X        | X     | X      |
| incremental           | Should bullets appear one at a time (on presenter mouse clicks)?                |      |     |      |     |     |    |        | X        | X     | X      |
| keep_md               | Save a copy of .md file that contains knitr output                              | X    |     | X    | X   | X   |    |        | X        | X     |        |
| keep_tex              | Save a copy of .tex file that contains knitr output                             |      | X   |      |     |     |    |        |          |       | X      |
| latex_engine          | Engine to render latex, "pdflatex", "xelatex", or "lualatex"                    |      | X   |      |     |     |    |        |          |       | X      |
| lib_dir               | Directory of dependency files to use (Bootstrap, MathJax, etc.)                 | X    |     |      |     |     |    |        | X        | X     |        |
| mathjax               | Set to local or a URL to use a local/URL version of MathJax to render equations | X    |     |      |     |     |    |        | X        | X     |        |
| md_extensions         | Markdown extensions to add to default definition or R Markdown                  | X    | X   | X    | X   | X   | X  | X      | X        | X     | X      |
| number_sections       | Add section numbering to headers  | X    | X   |      |     |     |    |        |          |       |        |
| pandoc_args           | Additional arguments to pass to Pandoc  | X    | X   | X    | X   | X   | X  | X      | X        | X     | X      |
| preserve_yaml         | Preserve YAML front matter in final document?                                   |      |     |      |     |     | X  |        |          |       |        |
| reference_docx        | docx file whose styles should be copied when producing docx output              |      |     | X    |     |     |    |        |          |       |        |
| self_contained        | Embed dependencies into the doc   | X    |     |      |     |     |    |        | X        | X     |        |
| slide_level           | The lowest heading level that defines individual slides                         |      |     |      |     |     |    |        |          |       | X      |
| smaller               | Use the smaller font size in the presentation?                                  |      |     |      |     |     |    |        | X        |       |        |
| smart                 | Convert straight quotes to curly, dashes to em-dashes, ... to ellipses, etc.    | X    |     |      |     |     |    |        | X        | X     |        |
| template              | Pandoc template to use when rendering file quarterly_report.html).              | X    | X   |      | X   |     |    |        |          | X     | X      |
| theme                 | Bootswatch or Beamer theme to use for page                                      | X    |     |      |     |     |    |        |          |       | X      |
| toc                   | Add a table of contents at start of document                                    | X    | X   | X    |     | X   | X  | X      |          |       | X      |
| toc_depth             | The lowest level of headings to add to table of contents                        | X    | X   | X    |     | X   | X  | X      |          |       |        |
| toc_float             | Float the table of contents to the left of the main content                     | X    |     |      |     |     |    |        |          |       |        |



# SYNTAXE STRICTE



Format de sortie après `output` :

```
output: html_document
```

Si on rajoute des "sub-options" :

- revenir à la ligne
- ajouter ":"
- revenir à la ligne
- indentation de 2 espaces
- rajouter la sub-option ":" + 2 espaces

## Rajouter un table des matières (toc) à l'HTML

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output: html_document  
---
```

## Rajouter un table des matières (toc) à l'HTML

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output: html_document  
---
```

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output:  
  html_document:  
    toc: TRUE  
---
```

Chaque format de sortie est paramétré indépendamment des autres

Pour les PDF: fixer la taille des figures a 8 x 8

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output:  
  html_document:  
    toc: TRUE  
  pdf_document:  
    fig_height: 8  
    fig_width: 8  
---
```

Pour html:

- Rajouter un toc flottant
- Choisir un theme

Pour pdf:

- Rajouter un toc



Pour html:

- Rajouter un toc flottant
- Choisir un theme

Pour pdf:

- Rajouter un toc



Ne pas oublier de renseigner

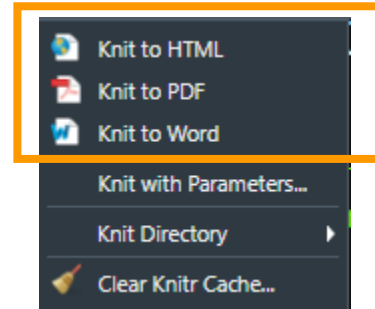
toc: TRUE

avant de mettre

toc\_float: TRUE

```
---  
title: "Exemple R Markdown"  
author: "Laurent Cauquil"  
date: "14 juillet 2023"  
output:  
  html_document:  
    toc: TRUE  
    toc_float: TRUE  
    theme: paper  
  pdf_document:  
    toc: TRUE  
    fig_height: 8  
    fig_width: 8  
---
```

On peut choisir le format de sortie dans l'interface RStudio



**Ctrl** + **shift** + **k**

Quand on "knite" en utilisant les raccourcis, c'est le dernier format de sortie utilisé qui est pris en compte

On peut également générer le document simultanément dans les différents formats avec la fonction `render()` du package `rmarkdown`

```
> rmarkdown::render("monfichier.Rmd", output_format = "all")
```

## Mise à jour de la date automatique

Dans les précédents exemples de YAML, la date est saisie en "dur"

Pour éviter d'avoir à la ressaisir à chaque fois (risque d'oublis ou d'erreur), on peut insérer la date courante directement, en exécutant un code R

On utilise des backquotes ``` pour indiquer qu'il y a du code à exécuter et on ajoute "r" pour spécifier le langage utilisé, puis enfin la fonction adéquate qui fait le job:

```
`Sys.Date()`
```

# Mise à jour de la date automatique

Dans les précédents exemples de YAML, la date est saisie en "dur"

Pour éviter d'avoir à la ressaisir à chaque fois (risque d'oublis ou d'erreur), on peut insérer la date courante directement, en exécutant un code R.

On utilise des backquotes ``` pour indiquer qu'il y a du code à exécuter et on ajoute "r" pour spécifier le langage utilisé, puis enfin la fonction adéquate qui fait le job:  
``Sys.Date()``.

```
---  
title: "Exemple R Markdown"  
subtitle: "Customisation avancée"  
author: "Laurent Cauquil"  
date: "`r Sys.Date()`"  
output: html_document  
---
```



Texte test avec des mots en *italiques*, en **gras**, mais aussi des exposants et des indices comme cette formule  $\text{H}_3\text{PO}_4^{2-}$

On verra plus tard comment souligner et ~~barrer~~

A vous de jouer



Texte test avec des mots en *italiques*, en **gras**, mais aussi des exposants et des indices comme cette formule  $\text{H}_3\text{PO}_4^{2-}$

On verra plus tard comment souligner et ~~barrer~~

A vous de jouer

```
Texte test avec des mots en *italiques*, en **gras**, mais aussi des  
exposants et des indices comme pour cette formule H~3~PO~4~^2-^
```

On utilise des séries de # pour définir la hiérarchie des titres

```
# First-level header  
## Second-level header  
### Third-level header
```

On peut rajouter une numérotation aux titres en rajoutant une sub-options dans le YAML

```
number_sections: TRUE
```

On peut ponctuellement supprimer cette numérotation en rajoutant {-} ou {.unnumbered} après un titre

```
# Preface {-}
```

Les listes sont définies par \*, - ou +,

```
- one item
* one item
- one item
  - one more item
  + one more item
  - one more item
```



- Sauter un ligne avant de commencer la liste
- Respecter les espaces entre le symbole et le texte

```
1. one item
2. one item
3. one item
  5. one more item
  10. one more item
  15. one more item
```



- Rajouter les chiffres avant le "." et sans espace
- Les chiffres se suivront obligatoirement au sein d'un même rang hiérarchique





Pour mettre en avant un petit paragraphe, on encadre le texte avec ">"

```
>  
REMARQUE  
>
```

Pour transformer un texte en format ligne de code, on l'encadre par des backquotes

```
`ligne de code`
```

Si cette ligne de code doit exécuter du code R, on rajoute un r en début

```
`r print("ligne de code")`
```

Le résultat de 2\*2 est : ``r 2*2``



Pour écrire grec, on rajoute des dollars \$

Markdown utilise la syntaxe LaTeX (<https://en.wikibooks.org/wiki/LaTeX/Mathematics>)

On encadre les formules **sans espace** avec un \$ ou avec \$\$ si on veut les centrer

```
$x = y$
```

```
$$\lim_{x \to \infty} f(x)$$
```

$$x = y$$
$$\lim_{x \rightarrow \infty} f(x)$$



R Markdown reconnaît les liens internet automatiquement (quand ils commencent par `http://` ou `https://`),

Ils seront affichés en bleu et le lien s'ouvrira si on clique dessus

Néanmoins pour gagner en lisibilité on peut associer un tag qui sera affiché à la place de l'adresse complète

```
Cliquer [ICI] (https://en.wikibooks.org/wiki/LaTeX/Mathematics)  
pour retrouver la page de la doc de LaTeX
```

Idem pour les images, en rajoutant ! au début et en spécifiant le chemin vers le fichier (ou l'adresse)

```
! [BLOB] (http://www.clker.com/clipart-green-blob.html)
```



Les renvois ou footnote sont automatiquement numérotés et ajoutés en fin de document

Ils sont insérés n'importe où dans le texte en utilisant les crochets et le caret

```
^[ceci est un renvoi]
```



On peut intégrer des citations dans le texte

R Markdown reconnaît la syntaxe des fichiers de bibliographies BibTeX (format exportable depuis les logiciels de bibliographies)

Toutes les références insérées dans le texte seront reportées en bas du document avec des renvois automatiques



Pour utiliser cette option, rajouter le champ "bibliography" ainsi que le chemin vers le fichier BibTeX dans le YAML

Puis le champ "link-citations" pour lier les références dans le texte avec le bas du document

```
title: " Exemple R Markdown "  
author: "Laurent Cauquil"  
date: "`r Sys.Date()`"  
output:  
  html_document:  
    toc: TRUE  
    toc_float: TRUE  
bibliography: ["biblio/biblio.bib"]  
link-citations: yes
```



Pour utiliser cette option, rajouter le champ `"bibliography"` ainsi que le chemin vers le fichier BibTeX dans le YAML

Puis le champ `"link-citations"` pour lier les références dans le texte avec le bas du document

Dans le script, une référence est introduite comme ceci (nom de l'auteur)

```
@grolemund_r_nodate
```

On rajoute des crochets pour mettre la référence entre parenthèses

```
[@grolemund_r_nodate]
```



Ligne de code R: "chunks"

# Ligne de code R: "chunks"



Le code R est inséré dans le script par l'intermédiaire de ce qu'on appelle des chunks

Un chunk est délimité par 2 séries de 3 backquotes, par défaut le code inséré dans un chunk est affiché, exécuté et le résultat affiché

```
` `` `{r}  
x = 2 + 2  
x  
` `` `
```

RStudio dispose de raccourcis pour ajouter rapidement des chunks dans le script:

➤ Bouton `insert --> R`

➤ Combinaison de touches

**Ctrl** + **Alt** + **i**



Il existe beaucoup d'options qui permettent de contrôler le comportement d'un chunk, que ce soit pour du calcul, des graphs, des tableaux ou du texte

Ces options sont insérées entre les accolades après le ``r,`` et sont de la forme ``tag = value``

Ce sont des options du package ``knitr``.

La documentation complète des options se trouvent ici : <https://yihui.name/knitr/options>



## Les labels

Il est possible (et même recommandé) de nommer chaque chunk (facilite la navigation dans RStudio)

Le label suit immédiatement le r après un espace, seules les options sont après une virgule

```
{r label_du_chunk, options}
```



Chaque label de chunk doit être unique  
Eviter également les points et les espaces



## Evaluation du code

- **eval**: boolean, le code est exécuté ou pas

boolean = TRUE ou FALSE

```
` `{r addition, eval = TRUE}  
x = 2 + 2  
x  
` `
```

Le code est exécuté et affiché

```
[1] 4
```



## Affichage des messages

Pour ne pas surcharger inutilement les sorties, notamment quand se chargent les packages

- **warning:**      booleen, affiche les warnings ou pas
- **error:**        booleen, affiche les errors ou pas
- **message:**      booleen, affiche les messages ou pas

```
library(dplyr)
```



## Affichage du code

- **echo:** booleen, affiche le code ou pas
- **collapse:** booleen, affiche le code et le résultat dans la même boîte

```
` `` {r}  
x = 2 + 2  
x  
` ``
```



## Formatage du texte de sortie

- **prompt:**      booléen, ajoute le prompt au début des lignes de codes (empêche les copier-coller direct dans la console, pas cool)
- **comment:**    string, définit le préfixe de commentaire (`##` en général)
- **highlight:**        booléen, ajoute la coloration syntaxique au code





## Affichage des résultats

- **include**: booleen, affiche le résultat ou pas. Le code est évalué dans tous les cas, les graphes générés mais pas affichés
  
- **results**:
  - "asis": affiche le résultat en style console R (brut)
  - "hide": cache le résultat (sauf si warning, message ou error)
  - "hold": regroupe et affiche tous les résultats du chunk à la fin de la boîte du code



## Pour les graphs

- **fig.width, fig.height**: dimensions du graphe en pouces
- **fig.dim**: raccourci pour **fig.width + fig.height** (ex: **fig.dim = c(5, 7)**)
- **fig.align**: "left", "center", "right", alignement en largeur
- **fig.show**:
  - "hold": regroupe et affiche tous les graphes à la fin de la boîte
  - "hide": génère les graphes mais ne les affiche pas
- **fig.keep**:
  - "first": affiche que le premier graphe
  - "last": affiche que le dernier graphe
- **dpi**: définition

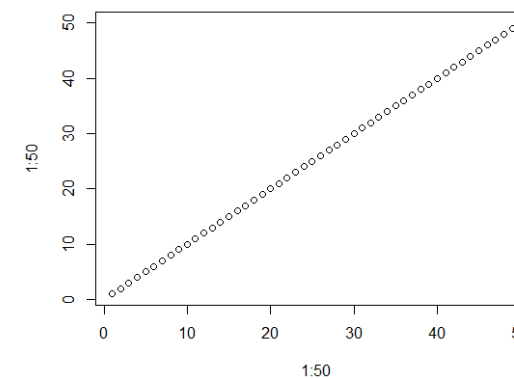
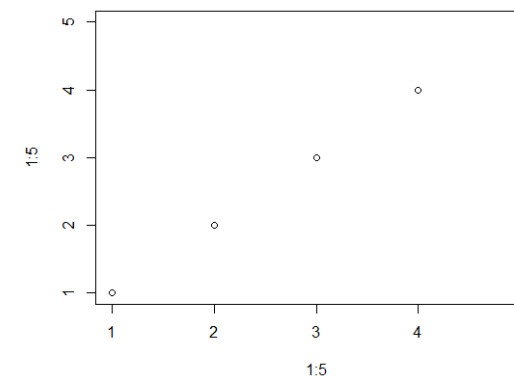
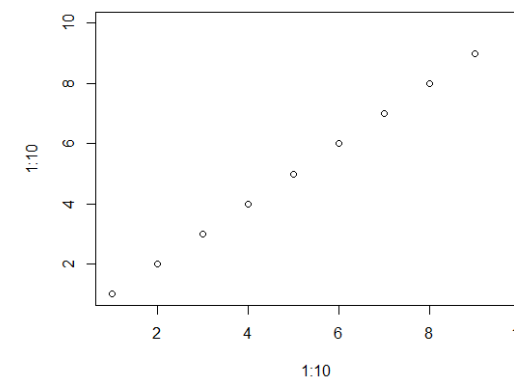
# Ligne de code R: "chunks"

# Les options



## Pour les graphs

Afficher 3 graphs à la suite avec une largeur de 6 et centrés dans la page



# Ligne de code R: "chunks"

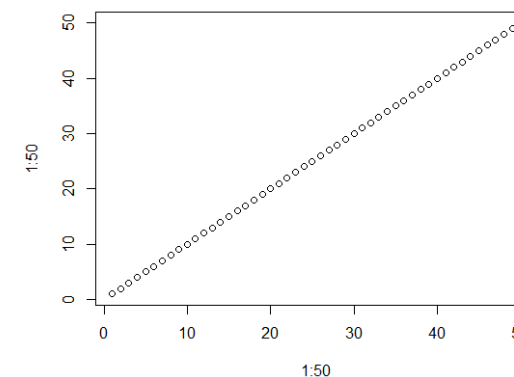
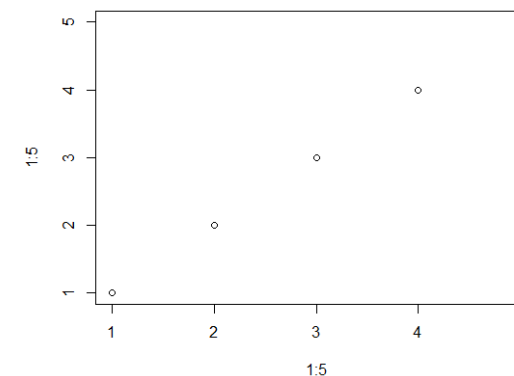
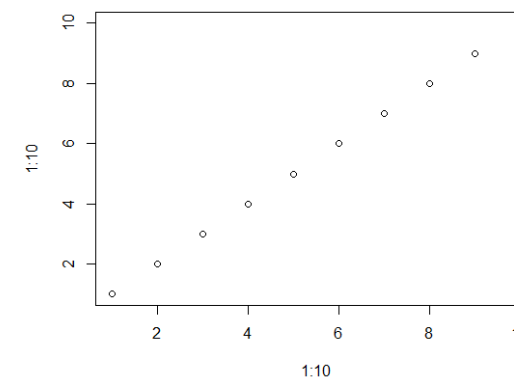
## Les options



Pour les graphs

Afficher 3 graphs à la suite avec une largeur de 6 et centrés dans la page

```
```{r, fig.align = "center", fig.width = 6,  
fig.show = "hold"}  
plot(1:10,1:10)  
plot(1:5,1:5)  
plot(1:50,1:50)  
```
```





## Insertion d'une image

On peut utiliser les chunks pour insérer une image plutôt que d'utiliser la syntaxe R Markdown (``![alt text or image title](path/to/image)``)

L'intérêt est d'avoir plus de contrôle sur le formatage

On utilise la fonction `include_graphics` du package `knitr`

```
```{r, fig.width = 6, fig.align = 'right'}
knitr::include_graphics('fig/blob.png')
```
```



## Chunk setup

On peut définir des options de chunk pour tous les chunk du document  
C'est le rôle du chunk de "setup"

Il est placé en début de document et suit la syntaxe suivante

`knitr::opts_chunk$set (options...)`

```
```${r setup, include=FALSE}  
knitr::opts_chunk$set(echo = TRUE, fig.align = "center")  
```
```



# Les tables

|                   | mpg  | cyl | disp | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4         | 21.0 | 6   | 160  | 110 | 3.90 | 2.620 | 16.46 | 0  | 1  | 4    | 4    |
| Mazda RX4 Wag     | 21.0 | 6   | 160  | 110 | 3.90 | 2.875 | 17.02 | 0  | 1  | 4    | 4    |
| Datsun 710        | 22.8 | 4   | 108  | 93  | 3.85 | 2.320 | 18.61 | 1  | 1  | 4    | 1    |
| Hornet 4 Drive    | 21.4 | 6   | 258  | 110 | 3.08 | 3.215 | 19.44 | 1  | 0  | 3    | 1    |
| Hornet Sportabout | 18.7 | 8   | 360  | 175 | 3.15 | 3.440 | 17.02 | 0  | 0  | 3    | 2    |

| car               | mpg  | cyl |
|-------------------|------|-----|
| Mazda RX4         | 21   | 6   |
| Mazda RX4 Wag     | 21   | 6   |
| Datsun 710        | 22.8 | 4   |
| Hornet 4 Drive    | 21.4 | 6   |
| Hornet Sportabout | 18.7 | 8   |
| Valiant           | 18.1 | 6   |
| Duster 360        | 14.3 | 8   |
| Merc 240D         | 24.4 | 4   |
| Merc 230          | 22.8 | 4   |
| Merc 280          | 19.2 | 6   |



# Les tables



## Tables en markdown

La syntaxe markdown permet de créer des tableaux directement dans le texte, mais ce n'est pas très pratique et le résultat n'est pas facilement personnalisable

Il faut jouer avec les `~`, les `|` et les `:` pour déterminer les différents champs

```
Auteur	Package	Année
Yixui Xie	knitr	2012
Hadley Wickham	ggplot2	2005
```

| Auteur         | Package | Année |
|----------------|---------|-------|
| Yixui Xie      | knitr   | 2012  |
| Hadley Wickham | ggplot2 | 2005  |

## Packages pour les tables

Il existe de nombreux packages qui permettent de formater les tables, ci-après une liste (non exhaustive) de packages:

Packages:

- DT: <https://rstudio.github.io/DT/>
- flextable: <https://davidgohel.github.io/flextable/>
- kableExtra (extension de kable): [https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome\\_table\\_in\\_html.html](https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html)
- htmlTable: <https://cran.r-project.org/web/packages/htmlTable/vignettes/tables.html>
- table1: <https://cran.r-project.org/web/packages/table1/vignettes/table1-examples.html>
- ztable: <https://cran.r-project.org/web/packages/ztable/vignettes/ztable.html>



Attention aux tables interactives pas prises en charge par les PDF



Fonction `datatable` sur un `dataframe`

```
` `{r, eval = TRUE, warning = FALSE}  
library(DT)  
datatable(iris)  
` ` `
```

Par défaut la page produite fournie:

- - Affichage paginé
- - Tri sur chaque colonne
- - Tri par recherche

<https://rstudio.github.io/DT/>

## Packages kableExtra (extension de kable)

A partir d'un **dataframe** crée un tableau avec la fonction **kable** et le personnalise avec des fonctions de **kableExtra** (?**kable\_styling**)

```
` `` {r}
suppressPackageStartupMessages (library (kableExtra) )
mtcars %>%
  head %>%
  kable () %>%
  kable_styling ()
` ``
```

<http://haozhu233.github.io/kableExtra/>

On peut combiner plusieurs options de l'argument `bootstrap_options`

```
` `` {r}
mtcars %>%
  head(3) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "bordered"),
                font_size = 12,
                full_width = FALSE,
                position = "float_right")
` ``
```

<http://haozhu233.github.io/kableExtra/>

## Packages kableExtra (extension de kable)

Personnalisation des lignes: **row\_spec**

- **row**: sélectionne les lignes (0, pour les en-têtes)
- **bold, italic, underline, strikeout**: booleen, format du texte
- **color, background**: couleur du texte et du fond
- **align**: left, right, center, justify
- **font\_size**
- **angle**

<http://haozhu233.github.io/kableExtra/>

# Les tables



## Packages kableExtra (extension de kable)

|                          | mpg         | cyl      | disp       | hp         | drat        | wt           | qsec         | vs       | am       | gear     | carb     |
|--------------------------|-------------|----------|------------|------------|-------------|--------------|--------------|----------|----------|----------|----------|
| <b>Mazda RX4</b>         | <b>21.0</b> | <b>6</b> | <b>160</b> | <b>110</b> | <b>3.90</b> | <b>2.620</b> | <b>16.46</b> | <b>0</b> | <b>1</b> | <b>4</b> | <b>4</b> |
| <b>Mazda RX4 Wag</b>     | <b>21.0</b> | <b>6</b> | <b>160</b> | <b>110</b> | <b>3.90</b> | <b>2.875</b> | <b>17.02</b> | <b>0</b> | <b>1</b> | <b>4</b> | <b>4</b> |
| <b>Datsun 710</b>        | <b>22.8</b> | <b>4</b> | <b>108</b> | <b>93</b>  | <b>3.85</b> | <b>2.320</b> | <b>18.61</b> | <b>1</b> | <b>1</b> | <b>4</b> | <b>1</b> |
| Hornet 4 Drive           | 21.4        | 6        | 258        | 110        | 3.08        | 3.215        | 19.44        | 1        | 0        | 3        | 1        |
| <b>Hornet Sportabout</b> | <b>18.7</b> | <b>8</b> | <b>360</b> | <b>175</b> | <b>3.15</b> | <b>3.440</b> | <b>17.02</b> | <b>0</b> | <b>0</b> | <b>3</b> | <b>2</b> |

<http://haozhu233.github.io/kableExtra/>

## Packages kableExtra (extension de kable)

```
```{r}
mtcars %>%
  head(5) %>%
  kable() %>%
  kable_styling(full_width = FALSE) %>%
  row_spec(row = 0,
           font_size = 15,
           color = "red",
           angle = -30) %>%
  row_spec(row = c(1:3,5),
           bold = TRUE,
           strikeout = TRUE,
           color = "green",
           background = "black")
```
```

<http://haozhu233.github.io/kableExtra/>



## Packages kableExtra (extension de kable)

Personnalisation des colonnes: **column\_spec**

- **column**: sélectionne les colonnes (1 correspond aux rownames)
- **width**: largeur des colonnes (rajouter l'unité)
- **border\_left, border\_right**: booleen, rajoute des bordures

<http://haozhu233.github.io/kableExtra/>

## Packages kableExtra (extension de kable)

|                  | mpg  | cyl | disp | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4        | 21.0 | 6   | 160  | 110 | 3.90 | 2.620 | 16.46 | 0  | 1  | 4    | 4    |
| Mazda RX4<br>Wag | 21.0 | 6   | 160  | 110 | 3.90 | 2.875 | 17.02 | 0  | 1  | 4    | 4    |
| Datsun 710       | 22.8 | 4   | 108  | 93  | 3.85 | 2.320 | 18.61 | 1  | 1  | 4    | 1    |

<http://haozhu233.github.io/kableExtra/>

## Packages kableExtra (extension de kable)

```
` ``{r}
mtcars %>%
  head(3) %>%
  kable() %>%
  kable_styling(full_width = FALSE) %>%
  column_spec(column = c(2,5),
              width = "3cm",
              color = "red",
              border_left = T, border_right = T)
` ``
```

<http://haozhu233.github.io/kableExtra/>

### Personnalisation des cellules avec les fonctions `cell\_spec`

Cette fonction est utilisée pour mettre en évidence le résultat d'un tri conditionnel sur les colonnes d'une table.



ATTENTION: cette fonction s'applique directement sur le **data.frame** avant de l'envoyer dans la fonction **kable()**

Il faut rajouter **escape = FALSE** en argument à la fonction **kable**

<http://haozhu233.github.io/kableExtra/>

# Les tables



## Packages kableExtra (extension de kable)

Dans l'exemple suivant, on crée 3 nouvelles colonnes avec la fonction **mutate** de dplyr:

➤ **car** =

➤ **mpg** =

➤ **cyl** =

| car               | mpg  | cyl |
|-------------------|------|-----|
| Mazda RX4         | 21   | 6   |
| Mazda RX4 Wag     | 21   | 6   |
| Datsun 710        | 22.8 | 4   |
| Hornet 4 Drive    | 21.4 | 6   |
| Hornet Sportabout | 18.7 | 8   |
| Valiant           | 18.1 | 6   |
| Duster 360        | 14.3 | 8   |
| Merc 240D         | 24.4 | 4   |
| Merc 230          | 22.8 | 4   |
| Merc 280          | 19.2 | 6   |

<http://haozhu233.github.io/kableExtra/>

# Les tables



## Packages kableExtra (extension de kable)

Dans l'exemple suivant, on crée 3 nouvelles colonnes avec la fonction **mutate** de dplyr:

- **car** = on récupère les rownames de la table
- **mpg** = affiche en bleu les "mpg" > 20 sinon affiche en rouge
- **cyl** = affiche en blanc, aligne au centre avec une rotation de 45°, la couleur de background de chaque cellule est une nuance de gris associée à une valeur de la "cyl" transformée en facteur.

Le data.frame ainsi créé est ensuite envoyé dans kable

| car               | mpg  | cyl |
|-------------------|------|-----|
| Mazda RX4         | 21   | 6   |
| Mazda RX4 Wag     | 21   | 6   |
| Datsun 710        | 22.8 | 4   |
| Hornet 4 Drive    | 21.4 | 6   |
| Hornet Sportabout | 18.7 | 8   |
| Valiant           | 18.1 | 6   |
| Duster 360        | 14.3 | 8   |
| Merc 240D         | 24.4 | 4   |
| Merc 230          | 22.8 | 4   |
| Merc 280          | 19.2 | 6   |

<http://haozhu233.github.io/kableExtra/>

## Packages kableExtra (extension de kable)

```
```{r}
mtcars[1:10, 1:2] %>%
  mutate(
    car = row.names(.),
    mpg = cell_spec(mpg, "html", color = ifelse(mpg > 20, "red", "blue")),
    cyl = cell_spec(cyl, "html", color = "white", align = "c", angle = 45,
                    background = factor(cyl, c(4, 6, 8),
                                       c("#666666", "#999999", "#BBBBBB")))
  ) %>%
  select(car, mpg, cyl) %>%
  kable(format = "html", escape = F) %>%
  kable_styling("striped", full_width = F)
```
```

<http://haozhu233.github.io/kableExtra/>

### Fonctions `text_spec`

La fonction `text_spec` peut être utilisée directement dans un RMarkdown pour formater du texte

On peut facilement appliquer les arguments suivant à du texte:

- `bold`, `italic`, `underline`, `strikeout`: booleen, format du texte
- `color`, `background`: couleur du texte et du fond
- `align`: left, right, center, justify
- `font_size`
- `angle`

<http://haozhu233.github.io/kableExtra/>



## Packages kableExtra (extension de kable)

If you want to **fast**, go **alone**. If you want to go **further** , go **together**

```
If you want to go `text_spec("fast", bold = T, color = "red")`, go  
`text_spec("alone", bold = T, color = "red")`.  
If you want to go `text_spec("further", bold = T, color = "green")`, go  
`text_spec("together", bold = T, color = "green")`
```

<http://haozhu233.github.io/kableExtra/>

### Sauvegarde d'une table

La fonction `save_kable` permet de sauvegarder une table dans 3 formats (.html, .png, .pdf et .jpeg)

```
```{r}
mtcars %>%
  head %>%
  kable() %>%
  kable_styling() %>%
  save_kable(file = "Kable_table.pdf", self_contained = T)
```
```

<http://haozhu233.github.io/kableExtra/>

## Packages flextable

| Sepal.Length | Sepal.Width  | Petal.Length | Petal.Width | Species       |
|--------------|--------------|--------------|-------------|---------------|
| 5.100        | <b>3.500</b> | 1.400        | 0.200       | <b>setosa</b> |
| 4.900        | <b>3.000</b> | 1.400        | 0.200       | <b>setosa</b> |
| 4.700        | <b>3.200</b> | 1.300        | 0.200       | <b>setosa</b> |
| 4.600        | <b>3.100</b> | 1.500        | 0.200       | <b>setosa</b> |
| 5.000        | <b>3.600</b> | 1.400        | 0.200       | <b>setosa</b> |
| 5.400        | <b>3.900</b> | 1.700        | 0.400       | <b>setosa</b> |
| 4.600        | <b>3.400</b> | 1.400        | 0.300       | <b>setosa</b> |
| 5.000        | <b>3.400</b> | 1.500        | 0.200       | <b>setosa</b> |
| 4.400        | <b>2.900</b> | 1.400        | 0.200       | <b>setosa</b> |
| 4.900        | <b>3.100</b> | 1.500        | 0.100       | <b>setosa</b> |

<https://davidgohel.github.io/flextable/index.html>

## Packages flextable

Le package `flextable` permet d'obtenir un rendu "LaTeX" dans des pages HTML

```
` `` {r}
suppressPackageStartupMessages (library (flextable) )
mtcars %>%
  head %>%
  flextable ()
` ``
```

<https://davidgohel.github.io/flextable/index.html>



Il comporte tout une panoplie de fonctions qui permettent de personnaliser les tables

```
` ``{r}
iris %>%
  head(10) %>%
  flextable %>%
  color(i = ~ Sepal.Length < 5,
        j = ~ Sepal.Length + Sepal.Width,
        color = "orange") %>%
  bold(j = c("Sepal.Width", "Species"), bold = TRUE)
` ``
```

<https://davidgohel.github.io/flextable/index.html>

## Packages flextable

Il comporte tout une panoplie de fonctions qui permettent de personnaliser les tables

| Sepal.Length | Sepal.Width  | Petal.Length | Petal.Width | Species       |
|--------------|--------------|--------------|-------------|---------------|
| 5.100        | <b>3.500</b> | 1.400        | 0.200       | <b>setosa</b> |
| 4.900        | <b>3.000</b> | 1.400        | 0.200       | <b>setosa</b> |
| 4.700        | <b>3.200</b> | 1.300        | 0.200       | <b>setosa</b> |
| 4.600        | <b>3.100</b> | 1.500        | 0.200       | <b>setosa</b> |
| 5.000        | <b>3.600</b> | 1.400        | 0.200       | <b>setosa</b> |
| 5.400        | <b>3.900</b> | 1.700        | 0.400       | <b>setosa</b> |
| 4.600        | <b>3.400</b> | 1.400        | 0.300       | <b>setosa</b> |
| 5.000        | <b>3.400</b> | 1.500        | 0.200       | <b>setosa</b> |
| 4.400        | <b>2.900</b> | 1.400        | 0.200       | <b>setosa</b> |
| 4.900        | <b>3.100</b> | 1.500        | 0.100       | <b>setosa</b> |

<https://davidgohel.github.io/flextable/index.html>

# CACHE

# CACHE



L'option de cache permet de sauvegarder le résultats d'un chunk pour éviter d'avoir à le recalculer à chaque knitr du script

C'est très utile lorsque les temps de calcul sont longs mais cette option doit être utilisée avec précaution

```
```{r, cache = TRUE}  
x = 2+2  
```
```



# CACHE



En effet, un chunk ne sera réévalué que lorsque le code à l'intérieur d'un chunk aura changé

Les risques d'erreurs arrivent si un chunk non modifié utilise un objet qui lui aura été modifié dans un chunk précédent, la modification de l'objet ne sera pas prise en compte dans le chunk "en cache"

```
` `` {r, cache = TRUE}  
x = 2+2  
` ``
```

# CACHE



Pour pallier ces accidents, il y a une option en-dessous du bouton "knit" qui permet de vider le cache

```
` `` {r, cache = TRUE}  
x = 2+2  
` ``
```

# ONGLETS

# ONGLETS



R Markdown permet d'insérer des onglets, très utile pour rendre un document plus facile à lire quand les sorties sont longues

Il faut rajouter l'attribut `{.tabset}` à la suite d'un titre

Les sous-titres suivants apparaitront dans des onglets différents:

```
## Section avec onglets {.tabset}

### Onglet n°1

plot(sin(seq(1,10,0.2)), ylim = c(-2, 2), type = "l")

### Onglet n°1

plot(sin(seq(1,50,0.2)), ylim = c(-2, 2), type = "l")
```

# Un peu d'HTML

# Un peu d'HTML

Le balisage HTML est directement utilisable dans le fichier source, même si on perd en lisibilité dans le script...

On peut utiliser directement du code HTML pour modifier ponctuellement le format d'un texte

La syntaxe de l'HTML est basée sur l'utilisation de balises d'ouvertures et de fermetures (`<>` et `</>`) qui encadrent le texte à modifier



## Un peu d'HTML

On commence par un cas particulier qui n'a besoin que d'une balise !

**<Br>** : retour à la ligne

# Un peu d'HTML

## Formatage de base sur du texte

```
<center>Texte centré</center>
```

```
<s>texte barré</s>
```

```
<font size = "6">Taille 6</font>
```

```
<font face="Arial">Change la police</font>
```

```
<font style="color:red">Change la couleur</font>
```





## Un peu d'HTML

### On peut cumuler les options

```
<center><s><font face="Courier New" size = "10" style="color:red">CUSTOMISATION</font></s></center>
```

### Format touche de clavier

```
<kbd>Alt</kbd> + <kbd>Shift</kbd>
```

# Rapports paramétrables

# Rapports paramétrables

Autre avantage des rapports R Markdown:

La possibilité de lancer un script de RMarkdown sur plusieurs jeux de données en ne changeant qu'un ou plusieurs paramètres mais sans rien changer au script !

Exemple détaillé de rapports paramétrés

<https://bookdown.org/yihui/rmarkdown/parameterized-reports.html>

# Rapports paramétrables



Les paramètres modifiables sont déclarés dans le YAML

L'exécution de R Markdown est lancée avec la fonction **render()** dans la console R

Elle prend en argument le fichier markdown et le ou les paramètre(s)

# Rapports paramétrables

## Tout d'abord le YAML

```
---  
title: "summary"  
author: "Laurent Cauquil"  
date: "23/08/2019"  
output: html_document  
params:  
  data: file.txt  
---
```

# Rapports paramétrables

Ensuite le chunk du R Markdown qui utilise le paramètre

```
```{r}
## Importation du fichier passé par le data de params
df <- read.table(params$data, h=T, sep = ";")

# Résumé de la table
summary(df)
```
```

Le script est sauvegardé dans: "rapport\_parametre.Rmd"

# Rapports paramétrables

## Création des jeux de données

On récupère les datasets `iris` et `cars` inclus dans R, que l'on exporte en ".csv"

```
```{r, eval = FALSE}  
write.table(iris, file = "iris.csv", sep = ";")  
write.table(cars, file = "cars.csv", sep = ";")  
```
```

## Rapports paramétrables

On lance l'exécution du R Markdown pour chaque fichier avec la fonction **render()** dans la console R.

```
rmarkdown::render(input = "rapport_parametre.Rmd",  
  params = list(data = "cars.csv"),  
  output_file = "rapport_cars")
```

```
rmarkdown::render(input = "rapport_parametre.Rmd",  
  params = list(data = "iris.csv"),  
  output_file = "rapport_iris")
```

On a généré 2 rapports **rapport\_cars.html** et **rapport\_iris.html** dans le répertoire courant



# PURL()

## PURL()

La fonction ``purl()`` (lancer dans la console R) permet d'extraire toutes les lignes de codes des chunks et de les sauvegarder dans un fichier.

L'argument ``documentation`` permet de choisir si on affiche ou non les labels et les options de chaque chunk.

- 0 : ne garde que le code
- 1 : affiche labels et options de chaque chunk

```
purl(input = "markdown_to_extract.rmd", output = "code_extracted.R",  
documentation = 1)
```



# RMARKDOWN VS NOTEBOOK

# RMARKDOWN vs NOTEBOOK



RStudio vous donne la possibilité de créer un R Markdown ou un Notebook, qu'est-ce qui les différencie ?

Rien lors de l'écriture du document excepté dans le YAML :

```
output: html_notebook
```

# RMARKDOWN vs NOTEBOOK



En revanche, c'est au moment de l'exécution que l'on voit la différence, pour les R Markdown on a un bouton ``knitr`` alors que pour un Notebook on a un bouton ``preview`` !

Lorsque on ``knitr`` un R Markdown, une session R est ouverte et tout le code est exécuté dans cet environnement puis la session est refermée

# RMARKDOWN vs NOTEBOOK



Un Notebook utilise l'environnement global (celui de la console R ouverte) et un fichier HTML est créé (avec toutes les résultats des chunks) puis mis à jour automatiquement à chaque modification

Ainsi, le résultat de chaque chunk (calculé auparavant) peut être visualisé à tout moment

Pour transformer un R Markdown en Notebook il suffit de changer le YAML et inversement

# ANNEXE



<https://bookdown.org/yihui/rmarkdown/>

<https://yihui.name/tinytex/>

<http://svmiller.com/blog/2016/02/svm-r-markdown-manuscript/>

<https://www.rstudio.com/resources/cheatsheets/#rmarkdown>

<https://yaml.org/spec/1.2/spec.html>

<https://pandoc.org/MANUAL.html#options>

<https://sumanmathmedicine.blogspot.com/2014/11/using-markdown-and-pandoc-for.html>

<https://dr-harper.github.io/rmarkdown-cookbook/>

<https://en.wikibooks.org/wiki/LaTeX/Mathematics>

<http://haozhu233.github.io/kableExtra/>

<https://davidgoheh.github.io/flextable/index.html>